IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received June 28, 2018, accepted August 5, 2018, date of publication August 21, 2018, date of current version September 7, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2866259

Dynamic Range-Only Localization for Multi-Robot Systems

YANJUN CAO[1], MENG LI[1,2], IVAN VOGOR[1], SHAOMING WEI[3], AND GIOVANNI BELTRAME[1]

[1]Computer and Software Engineering Department, Polytechnique Montréal, Montreal, QC H3C 3A7, Canada
[2]School of Information and Science Technology, Zhejiang Sci-Tech University, Hangzhou, China
[3]School of Electronic and Information Engineering, Beihang University, Beijing 100083, China

Corresponding author: Meng Li (lmbuaa@gmail.com)

This work was supported by the Natural Sciences and Engineering Research Council Strategic Partnership under Grant 479149-2015.

ABSTRACT The localization problem for multi-robot teams has been extensively studied with the goal of obtaining precise positioning information, such as required by a variety of robotic applications. This paper proposes a dynamic localization approach that exploits multiple robots equipped with range-only ultra-wideband sensors to create and maintain a common self-adaptive coordinate system. For 2-D localization, we use three robots with relative range measurements to build a global coordinate system. We recursively apply an extended Kalman filter, which results in accurate position estimates over time. We also propose a reconfiguration approach that prevents error accumulation from ultra-wideband sensors. The applicability of our approach is tested through a campaign of simulations, which show promising results.

INDEX TERMS Multi-robot, range-only localization, UWB, EKF.

I. INTRODUCTION

Multi–robot and swarm robotics systems have a high potential to solve various real–world challenges, such as underwater exploration, establishing network coverage, search and rescue missions, etc. [1]. A key advantage of swarm robotics in contrast to traditional approaches, is its decentralized architecture, which is most suited to highly dynamic and harsh environments, where it is hard to achieve sufficient system reliability and robustness using centralized systems. In addition, with a decentralized approach, solutions tend to be simpler and more flexible [2]. In this paper, we focus on a decentralized system architecture for the purpose of robot localization.

As one of the fundamental challenges in mobile robotics, robot localization has been extensively studied in the past [3]–[5], using various approaches, sensors, and for different application scenarios [6]–[9]. In general, based on their architecture, current localization methods can be classified in two main groups: *global/centralized* (e.g., using global positioning system, overhead cameras, motion capture systems, etc.) or *relative/decentralized* (e.g., using laser range finders, ultra–sonic sensors, onboard cameras, etc.).

Though a great number of publications address localization, there are still open challenges, specifically in the context of fully decentralized systems, as are swarms of robots.

In this paper, we propose a decentralized localization system for swarm robotics with ultra–wideband (UWB) ranging sensors in a two-dimensional plane. The selection of this particular sensor is due to its accuracy, low cost, and the fact that it is not limited to line–of–sight. However, localization using range–only information requires addressing of several issues.

First, the orientation of the robots needs to be estimated, and second, UWB range measurements become less accurate with distance, which requires to mitigate error accumulation. Our method addresses these issues in three stages: a) *initialization*, b) *localization*, and c) *reconfiguration* as illustrated by Fig. 1. In our approach, the swarm collectively creates and maintains a common coordinate system that is used for the localization of individual robots, while being continuously updated to minimize the position error coming from UWB measurements.

The primary goal of the *initialization* stage is to select three reference robots that are used to create a coordinate system. Initially, the swarm members elect the first robot, also called the leader. The choice of the leader is based on neighbor density, and the leader itself becomes one of the reference robots. After this step, the leader selects two additional robots which are then used to create a coordinate system. Finally, this coordinate system is shared with the other members in the swarm, which use the information to compute their position.[1] As the robots move to perform their tasks, their coordinates change and the distances can increase, thus building up

[1]The assumption used here is that all the robots within the swarm are able to communicate within the operating area.

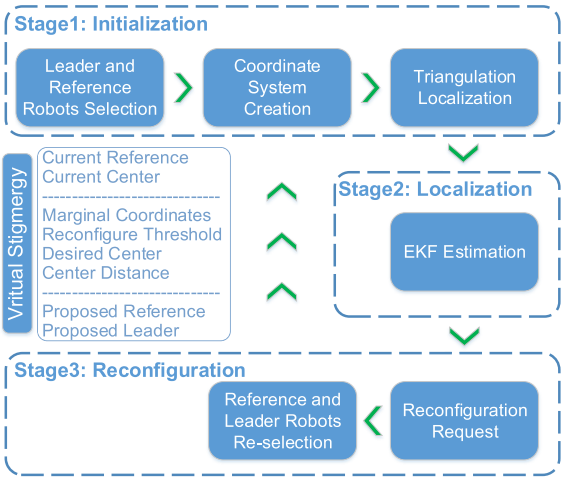

FIGURE 1. The system structure of dynamic range-only localization of multiple robots.

estimation errors. In practice, pure trilateration is not suitable for continuous localization. This issue of the dynamics of the system is handled in the localization stage.

In the *localization* stage, an extended Kalman filter (EKF) is applied for localization estimation. The EKF only requires the initial position of any given robot, and performs continuous localization using the common coordinate system. As the robots are moving, the UWB measurement error also builds up for the position of the reference robot, so this can potentially introduce additional error.

In the final stage, *reconfiguration*, we select a new set of reference robots based on the centroid of the swarm when the measurement uncertainty passes a given threshold, significantly reducing the estimation error. In the following sections, we describe all stages in detail, and show the accuracy measurements resulting from a simulation campaign.

To summarize, the contributions of this paper are:

(a) a dynamic self–adaptive approach for multi–robot localization based on range–only information,
(b) a method for determining a swarm central point and coordinate–system reconfiguration for the purpose of mitigating UWB sensor measurement error,
(c) a reconfiguration mechanism for the dynamic adaptation of the formation dynamics and to avoid error accumulation, and finally,
(d) an application of EKF to obtain accurate position estimation of moving robots using UWB sensors.

The rest of the paper is organized as follows: Section II makes a brief review of localization sensors, strategies, and estimation models; Section III illustrates the proposed localization system; in Section IV, we introduce the EKF and we detail the dynamic localization estimation; we evaluate the proposed approach is evaluated in simulation in Section V; finally, Section VI concludes the paper.

II. RELATED WORK

Localization is a well known and explored research subject: to present a structured and clear picture of the related work, this section is divided in three main topics in localization: sensors, strategies, and estimation models.

A. LOCALIZATION SENSORS

The application of traditional sensors, such as RGB or IR cameras, laser range finders, ultrasonic sensors, global positioning system (GPS), etc., in swarm robotics can be limiting. This is due to the fact that these sensors typically require heavy processing, or can only be used by a centralized system, or only work in certain types of environments.

Localization methods which use high-resolution RGB cameras usually rely on a vision subsystem which recognizes a set of QR markers (e.g. AprilTags [10]), which have a significant impact on processing resources [11]. To avoid this, researchers often use motion capture systems (e.g. Optitrack, Vicon [12], [13]) with a set of robots carrying physical IR markers that are unique for each robot. Although these methods are very precise, they are limited to a laboratory environment. In addition, the camera–marker solutions usually require centralized processing, which distributes the positions of the robots through a shared communication channel.

The use of GPS and derivatives (DGPS, RTK) is a convenient options, but its application is limited to a restricted set of environments. To ensure proper signal reception, it is often necessary to have a clear view of the sky, which mandates outdoor usage, with a *meter* scale precision. This is an ongoing issue for underwater, indoor, and robots for planetary exploration. In such environments, researchers often apply acoustic-based sensors or laser range scanners to use relative measures of the robot's environment for its localization. While there have been numerous successful applications of these, the decision between the two usually involves a tradeoff between precision and price. Since swarm robotics assumes tens, hundreds, or even more robots, this can be an issue. In addition, the usage of these sensors is limited to line–of–sight and requires significant processing capabilities.

Ultra-Wide Band (UWB) sensors are a promising solution for localization, considering the cost per unit, accuracy, and the available bandwidth [14]. They are advantageous for their inherent data transmission features, while not being restricted to line–of–sight. However, UWB sensors did not receive much attention in robotics due to their drawbacks in resolution and accuracy [15]. Recent works by González *et al.* [16], Hollinger *et al.* [17], Prorok and Martinoli *et al.* [14], [18], Prorok *et al.* [19] have shown that UWB sensors can be successfully used for localization, with accuracy around 5 cm. In particular, UWB sensors are suitable for swarm robotics as they can be used in a fully decentralized manner [20].

B. LOCALIZATION STRATEGIES

To successfully perform localization, the swarm of robots needs to agree on a reference coordinate system, which is used to transform all relative measurements into position data for each member of the swarm. A simple and straightforward approach is to use a fixed reference frame [21], [22].

This requires a set of beacons, which can either be static robots or external devices which are not part of the swarm. However, with this approach the general assumption is that a reference coordinate frame is known beforehand. This presents a major challenge for a swarm which is to be deployed in an unknown environment, with no access to global positioning, e.g. an underwater sensor array. Such scenario does not assume *a-priori* information about the environment nor a consensus on the reference frame. While this issue has been addressed in the past, current solutions require static swarm members that other robots use for localization (e.g., [23], [24]). Indelman *et al.* [25] use a probabilistic approach to achieve a consensus for the reference frame when dealing with spurious measurements (i.e., false negatives). In a recent publication, Shirazi and Jin [26] developed a localization method that requires a swarm of robots to be surrounded by several static robots which act as beacons to determine a common coordinate system. Using three robots, their algorithm provides coordinates for other robots within the team using relative trilateration. While this approach is promising, it requires that the robots be stationary in the localization stage. A good overview of traditional distributed estimation techniques, such as Non-Bayesian, Bayesian and Factor graphs is given by Wymeersch *et al.* [27]. However, in contrast to our approach, authors combine communication networks with a positioning system. In our paper, the communication network is used only for information sharing. The range sensors and non–linear Kalman filters are then applied to perform localization and infer position information.

C. ESTIMATION MODELS FOR LOCALIZATION

In multi-robot system, collaborative localization is a more robust approach to the localization problem [28]. While there are a variety of strategies, the prevailing methods use probabilistic models, which caught on in the early days of simultaneous localization and mapping (SLAM). Fox *et al.* [29] and Thrun [30].provided a general framework which used probabilistic methods, such as Monte–Carlo and Markov chains to localize robots, while keeping track of uncertainty. In such approaches, each robot is assigned a probabilistic cloud of particles representing its position, which is reduced when a robot detects a known feature in the environment, or another robot. Prorok *et al.* [31] proposed a similar approach for multi–robot localization using a range and bearing sensor. Their goal was to minimize the overall complexity of the particle filter based approach by defining reciprocal sampling which allowed to reduce the number of necessary particles. Luft *et al.* [32] recently proposed a fully decentralized localization algorithm based on the EKF. The algorithm tracks inter–robot correlations and it does not require measurement storage. Their method focuses on localization, assuming limited communication to neighboring robots. In this paper, we adopt a similar approach using EKF filters while a relative coordinate system is built based on range–only UWB sensors.

III. RANGE-ONLY SELF-ORGANIZED LOCALIZATION

As a team of robots equipped with UWB sensors is deployed within an unknown area, their localization becomes a challenge. These robots need to use only range information to create a common coordinate system in which every robot can estimate its position and that of its team members. To be applicable in the real world, the entire procedure should be fully automated, self–organized, and maintained in time so that the robots can seamlessly perform their tasks.

To complete this challenge, we propose a localization procedure divided into three consecutive stages: a) localization system initialization, b) localization and state estimation using an EKF, and c) the localization system dynamic reconfiguration.

A. STAGE 1: LOCALIZATION SYSTEM INITIALIZATION

As initial state, we assume that all members of a multi–robot team are scattered in an unknown region. To successfully perform localization, the robots first need to agree on a common reference frame, i.e. robots need to develop a common coordinate system. Since we are dealing with range–only information, the approach to developing a common coordinate system in this paper requires three robots, namely a *leader* and two *reference robots*. The imaginary line connecting the reference robots then represents the x–axis, while the imaginary line crossing through the leader robot as an orthogonal projection to x–axis, represents the y–axis. After this, a common coordinate system can be shared between all robots, which can infer their positions using trilateration.

In general, the robots in the multi–robot team are classified into three categories:

- reference robots – after initialization, each reference robot keeps estimating its position. In addition, robots that have been localized can be added into a shared table as reference robots;
- non-reference robots – use the position of reference robots to estimate an initial position based on trilateration. The non-reference robots localized by the trilateration become reference robots;
- marginal robots – represent the robots at the boundary of the current reference frame (further described in the following).

The information between robots is shared by means of a bio–inspired distributed consensus system called Virtual Stigmergy. The *Virtual stigmergy* can be seen as a tuple space shared between all the robots in the swarm via a communication medium [33]. We refer to this tuple space as the virtual stigmergy table (VST). This means that, once a robot stores some data into the VST, this information will gradually propagate until it is shared by all swarm members within communication range, although not instantly. For this work, we described the robots' behavior with a domain–specific programming language for swarm robotics called Buzz, in which the VST and its propagation are built-in [34].

Using the VST, all robots reach a consensus on which robot is elected leader, along with two additional reference robots.

To clarify further, consider the initialization phase, which is divided in three tasks: leader robot election, reference robot selection, and finally coordinate system creation and propagation.

1) LEADER ROBOT ELECTION

The election for the leader robot is by bidding on neighbor density. Therefore, the robot which has the most neighbors within a given range will be selected as a leader. Such choice is based on two reasons: a) the range of the UWB is limited, meaning that picking any random robot would not guarantee that it will have reference robots within its range, and also b) selecting robots which are clustered closer together reduces the initial measurement error of the UWB (as it is increased with distance). We would emphasize that in the initialization stage, we develop an initial (possibly sub-optimal) coordinate system that is improved in the next stages.

Initially, each robot obtains the number of its neighbors and offers it as a bid in the VST. This information is propagated between the robots, and once the robot with most neighbors becomes known, it is selected as a leader, by its unique identifier (ID). Fig. 2 shows this step, and illustrates several differently colored candidates. In this example, the red candidate has the highest number of neighbors and it is selected as a leader.

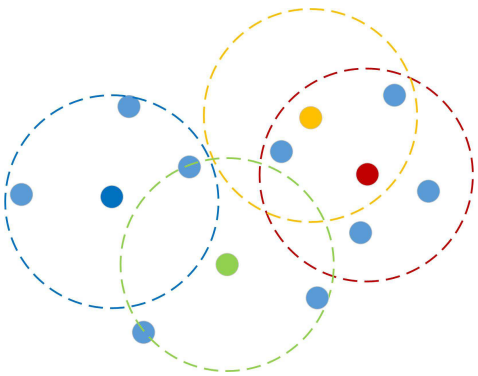

FIGURE 2. Leader election based on distributed consensus.

2) REFERENCE ROBOTS SELECTION

Once the swarm determines the leader, it needs two more reference robots to create a common 2D coordinate system. The reference robots are selected based on the following criteria: a) a robot is within the UWB measurement range of the leader, b) robot is the closest to the leader, so that it has lower measurement error from the UWB sensor. Fig. 3 illustrates two selected robots (a, b, marked in yellow), which are used in the following step to create the coordinate system.

To clarify the proposed approach, Algorithm 1 (lines 1–27) provides in-depth details. The algorithm starts by initializing all the necessary local variables (lines 1–3); where *leader_id* stores the identifier of the elected leader, *reference_robot* stores the selection of reference robots, *leader_election* is a flag which indicates whether the procedure of electing the leader is finished, and finally, *bid* contains the number of neighbors of a current robot within a certain range (using a UWB sensor).

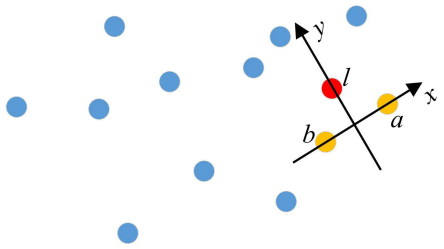

FIGURE 3. Reference robots (red and yellow dots) create coordinate system.

Algorithm 1: Coordinate System Initialization Phase

input : Virtual stigmergy table (VST)
output: Virtual stigmergy table with robots l, a, b and a coordinate system.

```
1  leader_id, reference_robot(.a, .b) ← ∅;
2  leader_election ← true;
3  bid ← neighbors.count();
4  while leader_election do
5      if bid < VST.get_leader_bid() then
6          leader_id ← VST.get_leader_id();
7      else
8          VST.get_leader_bid() ← bid;
9          VST.get_leader_id() ← id;
10         leader_id ← id;
11     end
12     if VST.leader_barrier() then
13         leader_election ← false
14     end
15 end
16 if id == leader_id then
17     robots_list =
       neighbors.foreach().sort_by_dist();
18     for robot_i, robot_{i+1} from robots_list do
19         if check_in_line() then
20             robots_list.delete(robot_i)
21         end
22     end
23     reference_robots.a ← robots_list[0];
24     reference_robots.b ← robots_list[1];
25     VST.reference_robots.a ← reference_robots.a;
26     VST.reference_robots.b ← reference_robots.b;
27 end
28 if id == leader_id then
29     create_coordinate_system();
30 end
```

Once they reach the main while loop (line 4), robots read the current highest *bid*, i.e. number of neighbors that is shared via Virtual Stigmergy, using the `VST.get_leader_bid()` operation. If the current bid is lower than the one that a current

robot has, it will swap its value, along with its identifier in the VST. This process is continuously repeated until a *barrier* is triggered.

In this context, a barrier refers to a mechanism which halts the further execution of the program until all robots reach consensus. In this case, they wait for everyone to have a chance to compare its bid. The barrier is a direct implementation of consensus among robots in the swarm. Essentially, it uses the VST, and a swarm table (a construct which contains the information about all swarm members in a decentralized manner) [33]. Moreover, once the operation `VST.`*leader_barrier*() returns True, the leader election is complete. Afterwards (lines 16–27), if the current robot is the leader robot, it sorts its neighbors based on distance and makes a *robot_list*. Every couple of robots in the list is checked not to be forming a line with the leader by function *check_in_line*. Finally, the leader selects its two closest non-colinear neighbors as references and writes their identifiers into the VST. Also, (lines 28–30) the same robot constructs the common coordinate system with the *create_coordinate_system*(), which uses the barrier mechanism to wait for all robots to confirm that the system was received. The details on the coordinate system creation are given in the following.

3) BUILDING A COMMON COORDINATE SYSTEM

After electing the leader along and two companion reference robots, we can create a coordinate system. Although our approach was partially inspired by the work of Shirazi and Jin [26], we propose a more intuitive way of creating a common coordinate system, which is performed by the leader robot. Fig. 3 also illustrates that the reference robots a and b lie on the x–axis, while the leader robot l lies on a positive y–axis, which is an orthogonal projection on the x axis. We then define the coordinate system using the following equations:

$$\begin{aligned} z_{la}^2 &= x_a^2 + y_l^2 = (d_{la} + v_{la})^2, \\ z_{lb}^2 &= x_b^2 + y_l^2 = (d_{lb} + v_{lb})^2, \\ z_{ab}^2 &= x_a^2 + x_b^2 = (d_{ab} + v_{ab})^2, \end{aligned} \tag{1}$$

where z_{la}, z_{lb}, and z_{ab} represent the range measurements from robots l to a, from l to b, and from a to b, respectively. d_{la}, d_{lb}, and d_{ab} represent the true distance between two robots, while v_{la}, v_{lb}, and v_{ab} are the corresponding measurement noise, which are subsequently corrected in the filter. From the above equations, the coordinates of the leader and the two referent robots are solved in the following form; $(0, y_l)$, $(x_a, 0)$, and $(x_b, 0)$, respectively, which are determined by the leader using the following equations:

$$\begin{aligned} x_a &= \frac{1}{2}\left(z_{ab} + \frac{z_{la}^2 - z_{lb}^2}{z_{ab}}\right), \\ x_b &= \frac{1}{2}\left(\frac{z_{la}^2 - z_{lb}^2}{z_{ab}} - z_{ab}\right), \\ y_l &= \sqrt{z_{la} - {x_a}^2} = \sqrt{z_{lb} - {x_b}^2}. \end{aligned} \tag{2}$$

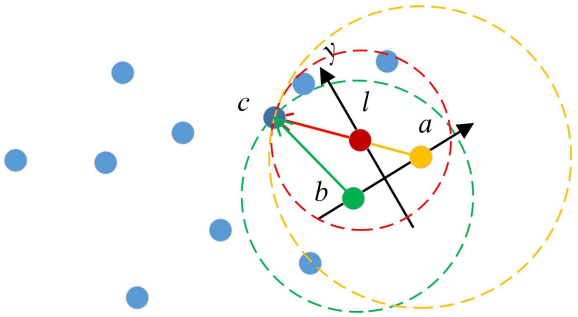

FIGURE 4. Localization of other robots within the coordinate system.

The leading robot l shares this information, along with the corresponding robot identifier using the VST. After the propagation of the VST data, all robots share the common coordinate system which is used to calculate their positions and positions of their neighbors. Fig. 4 shows how a robot c performs trilateration, i.e. signal intersection from the center points of the leader robot l (dashed red circle), and reference robots a (green dashed circle) and b (orange dashed line).

Only during this initial phase while creating the common coordinate system, we require the robots to be stationary. Afterwards, all robots execute the same EKF localization algorithm and perform collaborative localization.

Comparing to other similar work by Kurazume *et al.* [23], Corenejo and Nagpal *et al.* [24], or Shirazi *et al.* [26] , this is a significant improvement, as methods proposed in their work also require some robots to be static during the entire localization phase. Also, our approach does not require that the first three reference robots cover the distribution of the entire swarm. If the reference nodes are not within UWB range, neighbors who are already localized are used for trilateration, as implemented in our simulations. After the initialization, the robots use EKFs to estimate their positions with UWB measurements from their neighbors.

B. STAGE 2: LOCALIZATION AND ESTIMATION WITH THE EKF

After the initialization is completed, the initial positions of all robots are fed to the EKF to continuously estimate their position. Further details on the EKF is provided in Section IV.

This concludes the process of creating a common coordinate system and performing the initial localization. As the robots continue to perform their tasks, the UWB error contributes to the overall localization error of each robot [35], so the following task is to mitigate the error by optimizing the selection of robots that are used to create a new and more suitable coordinate system.

C. STAGE 3: DYNAMIC COORDINATE SYSTEM RECONFIGURATION

Having initialized a common coordinate system, in this stage the swarm is able to use their individual positions to mitigate erroneous measurements by the UWB, which are increasing

with the distance. In particular, robots are now able to reduce this error by reelecting a new leader and new reference robots, that are roughly equally distant from all other members, i.e. those that are closest to the centroid of their distribution within the operating area, rather than just based on the neighbor density as in the initial stage.

We refer to this stage as the reconfiguration, and it takes place in two cases: immediately after the initialization of the initial coordinate system and b) whenever the distribution of the robots has drastically changed from the previous configuration.

The dynamic reconfiguration of the coordinate system involves three steps: marginal robot selection, reconfiguration triggering, and coordinate system reconfiguration.

1) MARGINAL ROBOT SELECTION

While in operation, robots continuously move and consequently their pose estimation changes. Therefore, the members of the multi–robot team need to keep track of their distribution, and in particular the change of this distribution since the previous reconfiguration.

For this purpose, we developed an approach by which robots continuously select marginal robots, i.e. robots closest to the edge of their distribution. This is performed in a similar way as the bidding for the leader. However, in this case, using the VST, robots compare their minimum and maximum x and y coordinates with the goal of determining the top-, bottom-, left-, and right-most robots. Note that we assume that initially all robots are randomly distributed within a 2D Euclidean plane. Therefore, each robot writes its position in VST only if its position is further away than what already written in the VST, hence becoming a *marginal* robot.

The position of marginal robots is then used to calculate the centroid dC of a tetragon bounded by their position. This centroid position is assumed to be the center point of the distribution of robots, and it is used as a candidate point around which the new leader should be found. Such tetragon, along with the centroid, is illustrated in Fig. 5. The point cC represents the origin of a current coordinate system.

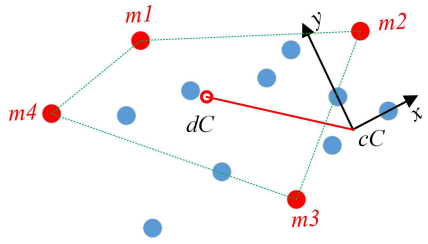

FIGURE 5. Selection of marginal robots (red dots).

2) RECONFIGURATION REQUEST TRIGGER

The robots continuously calculate the distance between the current origin cC and the candidate origin dC, and if its value is beyond a given threshold, it triggers a reconfiguration request.

3) COORDINATE SYSTEM RECONFIGURATION

Robots are notified that the reconfiguration is about to take place via the VST and go into the same process as in the initialization step described in Section III-A.1.

Namely, robots start the selection process for the leader robot l, which is followed by the selection of two new reference robots a and b, as depicted in Fig. 6. Note, however, that the leader is selected as the robot closest to the calculated dC, as in this scenario the bidding criterion is the positional information of dC, instead of the neighbor density.

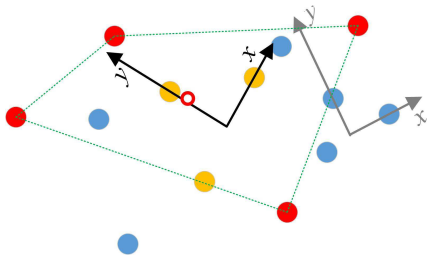

FIGURE 6. Reconfiguration of coordinate system.

Furthermore, once the leader and reference robots are selected, the following procedure of coordinate system propagation and continuous localization are the same as previously described in Section III-A.3. Only after the reconfiguration process is complete, robots start to use the new coordinate system, along with the new position within that reference frame.

Algorithm 2 shows the details of the proposed approach for continuous coordinate system reconfiguration. The input of the algorithm is the VST and the *position* of the robot.

The algorithm starts by initializing local variables: VST.*marg_robots* which is used to store current marginal robots, and *new_ref _robots*(*.leader*, *.a*, *.b*) which stores the new set of reference robots (i.e. leader and two references). The while loop (lines 4–8) uses the barrier mechanism through which we ensure that all robots reach a consensus on marginal robots. Specifically, all robots compare their positions with the current position of marginal robots using the *compare* operation. This operation returns *True* if and only if the minimum and maximum coordinates are such that a marginal robot needs to be replaced, which is performed by the *update* operation. Once the barrier is reached, the following step (line 9) calculates the centroid of the tetragon bound by the marginal robots, and this value is stored in VST.*marg_centroid*. After this, all robots are aware of the centroid position and are able to obtain reference robots. If the threshold is triggered (line 10), the closest robot (again agreed through the VST, line 11) obtains the list of all their neighbors sorted by distance (line 12). The first three robots are selected as reference robots (lines 13–15). Finally (lines 17–20), the new coordinate system is shared by the leader through the VST.

Algorithm 2: Coordinate System Reconfiguration

input : VST, *position*
output: VST with all robots sharing the new coordinate system

```
VST.marg_robots ← ∅ ;
reconfiguration_distance ← inf .;
new_ref _robots(.leader, .a, .b) ← ∅;
while VST.marg_robots_barrier() do
    if compare(position, VST.marg_positions) then
        update(VST.marg_robots, position);
    end
end
VST.marg_centroid = center(VST.marg_robots);
if dist(VST.marg_centroid, VST.cur_centroid) >
 VST.marg_rob.threshold then
    if min(dist(position, VST.marg_rob.centroid))
    then
        robots_list =
        neighbors.foreach().sort_by_distance();
        new_ref _robots.leader ← robots_list[0];
        new_ref _robots.a ← robots_list[1];
        new_ref _robots.b ← robots_list[2];
    end
    VST.ref _robots ← new_ref _robots;
    VST.leader_id ← new_ref _robots.leader;
    if id == VST.ref _robots.leader : then
        VST.coord_system=create_coord_system();
    end
end
```

The proposed reconfiguration process is running continuously to minimize the erroneous measurements by the UWB minimal on average, with respect to the distribution of robots.

IV. DYNAMIC LOCALIZATION ESTIMATION WITH EKF

In our system, every robot continuously estimates its location in the coordinate system after initialization or reconfiguration. In the beginning, a robot only needs the initial coordinates provided by trilateration, which is kept updated with an EKF estimator. Other high-order Kalman filters could also be considered as future work, when dealing with very high nonlinearities [36].

In this paper, we assume that a team of mobile robots is operating in a 2D Euclidean plane and the position tracking is based on range-only measurements, which is a marked difference from state-of-the-art scenarios using bearing information. It is reasonable to assume that the initial position of each robot can be accurately computed based on the range measurements in the initial frame of reference [37]. Assuming that robot i can move in 2D with speed v_i, in each time–step it can only move within a circular region centered in its position. Note that since the motion of a target robot can be reliably predicted for the next time step only, our objective is to locate and track the position of robots at consecutive time steps.

A. STATE ESTIMATION

1) EXTENDED KALMAN FILTER

As physical processes and/or observation models are generally nonlinear, we cannot directly apply a linear Kalman filter. To overcome this issue, the standard approach is to use a linearized EKF model, obtained by continuously linearizing models before applying estimation techniques [38]–[40].

Let $\mathbf{x}_i(k) = [x_i(k)\ \dot{x}_i(k)\ y_i(k)\ \dot{y}_i(k)]^T$ be the i-th robot state at time t_k. $(x_i(k), y_i(k))$ is the i-th robot position in Cartesian coordinates and the dot notation indicates differentiation with respect to time. Considering a random walk, the motion model can be given in the following form:

$$\begin{aligned} \mathbf{x}_i(k) &= \mathbf{F}_k \mathbf{x}_i(k-1) + \mathbf{w}_k, \\ \mathbf{F}_k &= \mathbf{I}_2 \otimes \begin{bmatrix} 1 & T_k \\ 0 & 1 \end{bmatrix}, \end{aligned} \tag{3}$$

where $\mathbf{w}_k$ is the process noise, which we assume white Gaussian, $T_k = t_k - t_{k-1}$, $\mathbf{I}_2$ is a 2×2 identity matrix, and $\otimes$ denotes the Kronecker product. At time t_k, a robot produces range-only measurements to reference robots.

In most practical navigation applications, a reference trajectory does not exist beforehand. Therefore, the EKF uses the current estimated state at each time step k as a linearization point. If the filter operates properly, the linearization error around the estimated solution can be maintained at a reasonably small value [40], [41].

Generally, the EKF algorithm can be described as:

$$\begin{aligned} \hat{\mathbf{x}}_{k|k-1} &= F_k \hat{\mathbf{x}}_{k-1}, \\ \mathbf{P}_{k|k-1} &= \mathbf{F}_{k-1}\mathbf{P}_{k-1}\mathbf{F}_{k-1}^T + \mathbf{G}_{k-1}\mathbf{Q}_{k-1}\mathbf{G}_{k-1}^T, \\ \mathbf{K}_k &= \mathbf{P}_{k|k-1}\mathbf{H}_k^T \left(\mathbf{H}_k\mathbf{P}_{k|k-1}\mathbf{H}_k^T + \mathbf{M}_k\mathbf{R}_k\mathbf{M}_k^T\right)^{-1}, \\ \hat{\mathbf{x}}_k &= \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k\left(\mathbf{z}_k - \mathbf{h}\left(\hat{\mathbf{x}}_{k|k-1}\right)\right), \\ \mathbf{P}_k &= \left(\mathbf{I} - \mathbf{K}_k\mathbf{H}_k\right)\mathbf{P}_{k|k-1}, \end{aligned} \tag{4}$$

where $\hat{\cdot}$ stands for an estimate, $\mathbf{P}$ is the state covariance matrix, $\mathbf{Q}$ the process noise covariance matrix, $\mathbf{R}$ the measurement noise covariance matrix, $\mathbf{F}$ the state transition matrix, $\mathbf{K}$ the Kalman gain, $\mathbf{H}$ the observation matrix, $\mathbf{G}$ the Jacobian matrix with respect to process noise, and $\mathbf{M}$ the Jacobian matrix with respect to measurement noise. Note that the state-transition function is linear, and the observation function $\mathbf{h}(\cdot)$ is non–linear.

B. MEASUREMENT MODEL

At t_{k+1}, robot-i measures its range $d_j(k+1)$ to the reference robots, $j = 1, \cdots, N$, where N is the number of reference robots. Therefore the measurement equation is:

$$\mathbf{z}(k+1) = \begin{bmatrix} d_1(k+1) \\ \vdots \\ d_N(k+1) \end{bmatrix} + \begin{bmatrix} \mathrm{w}_{d_1}(k+1) \\ \vdots \\ \mathrm{w}_{d_N}(k+1) \end{bmatrix}, \quad i = 1, \cdots, N. \tag{5}$$

$$d_j(k+1) = \sqrt{\Delta x_{i,j}^2(k+1) + \Delta y_{i,j}^2(k+1)}, \tag{6}$$

where $\Delta x_{i,j}(k+1) = x_i(k+1) - x_j(k+1)$ and $\Delta y_{i,j}(k+1) = y_i(k+1) - y_j(k+1)$ are the relative positions of i–th and the j–th reference robot, respectively, expressed in current coordinates. Note also that

$$\mathrm{w}(k+1) = [\mathrm{w}_{d_1}(k+1), \cdots, \mathrm{w}_{d_N}(k+1)]^T \quad (7)$$

is the noise in robot i's measurements, which we assume zero–mean white Gaussian noise with covariance

$$R_i(k+1) = E[\mathrm{w}_{d_i}(k+1)\mathrm{w}_{d_i}^T(k+1)] = diag(\delta_{d_i}^2) \quad (8)$$

and

$$\mathbf{R}(k+1) = E[\mathrm{w}(k+1)\mathrm{w}^T(k+1)] = diag(R_i(k+1)). \quad (9)$$

The measurement Equation (5) is a nonlinear function of the state variable $\mathbf{x}_i$. The measurement-error equation, obtained by linearizing Eq. 4 is

$$\begin{aligned}\tilde{\mathbf{z}}(k+1|k) &= \mathbf{z}(k+1) - \hat{\mathbf{z}}(k+1|k) \\ &\approx H_{k+1}\tilde{\mathbf{x}}_i(k+1|k) + \mathrm{w}(k+1), \end{aligned} \quad (10)$$

where

$$\begin{aligned}\hat{\mathbf{z}}(k+1|k) &= [\hat{z}_1^T(k+1|k), \hat{z}_2^T(k+1|k), \cdots, \hat{z}_N^T(k+1|k)]^T \\ \hat{z}_i^T(k+1|k) &= [\hat{d}_i(k+1|k)]^T \\ \hat{d}_j(k+1|k) &= \sqrt{\Delta\hat{x}_{i,j}^2(k+1|k) + \Delta\hat{y}_{i,j}^2(k+1|k)} \\ \Delta\hat{x}_{i,j}(k+1) &= \hat{x}_i(k+1|k) - x_j(k+1) \\ \Delta\hat{y}_{i,j}(k+1) &= \hat{y}_i(k+1|k) - y_j(k+1) \\ \tilde{\mathbf{x}}_i(k+1|k) &= \mathbf{x}_i(k+1) - \hat{\mathbf{x}}_i(k+1|k). \end{aligned}$$

Note that the Jacobian matrix of measurement is given by the following expression:

$$\begin{aligned} H_{d_j,k+1} &= \frac{\partial \mathbf{h}_{d_j,k+1}}{\partial \mathbf{x}_{k+1}}, \\ \mathbf{h}_{d_j,k+1} &= [d_1(k+1), \cdots, d_N(k+1)]. \end{aligned}$$

One can observe that the measurement Equation (6) is nonlinear and its first derivative exist, which justifies the use of the EKF.

V. SIMULATION RESULTS

This section demonstrates and evaluates the proposed approach separately, with ARGoS [42] and Matlab simulations. Matlab was mainly used as a proof–of–concept, while ARGoS is a multi–physics simulator, which provides a concrete step towards the implementation of the algorithm on a team of robots. In addition, ARGoS natively supports Buzz which, as previously mentioned, implements the VST that is essential for this work.

A. SIMULATIONS OF SYSTEM INITIALIZATION AND RECONFIGURATION

Consider a swarm of robots as shown in Fig. 7. Here, robots are involved in the bidding process to select a leader robot. Each robot has a range within which it can detect its neighbors, and for this simulation it is set to 2 m (one-floor rectangle represents one meter). All robots apply Algorithm 1

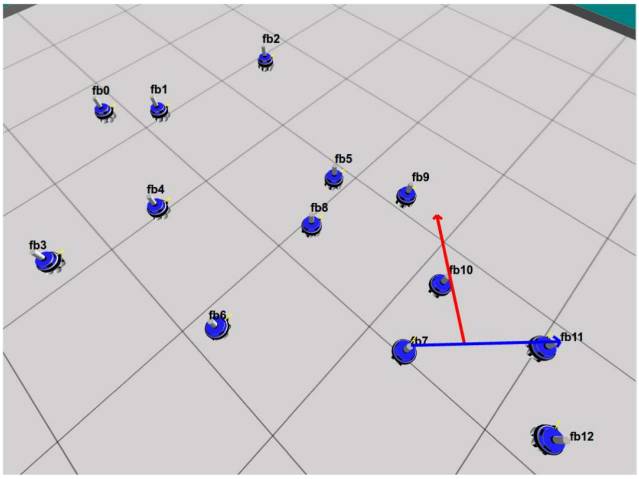

FIGURE 7. System initialization stage: three robots are selected as reference robots to build a coordinate system.

and achieve the consensus that the robot `fb10` is the initial leader. According to the proposed algorithm, its two closest neighbors, robots `fb7` and `fb11` are selected as reference robots which it uses to create the initial coordinate system (which is also illustrated in Fig. 7).

Suppose that after the initialization, the reconfiguration process is triggered and a new coordinate system needs to be constructed and shared with all robots. Since in this phase, all robots are aware of their positions in a previously constructed coordinate system, four marginal robots are selected via distributed consensus. As shown in Fig. 8 these are `fb0, fb2, fb11, fb12`, which together form a tetragon.

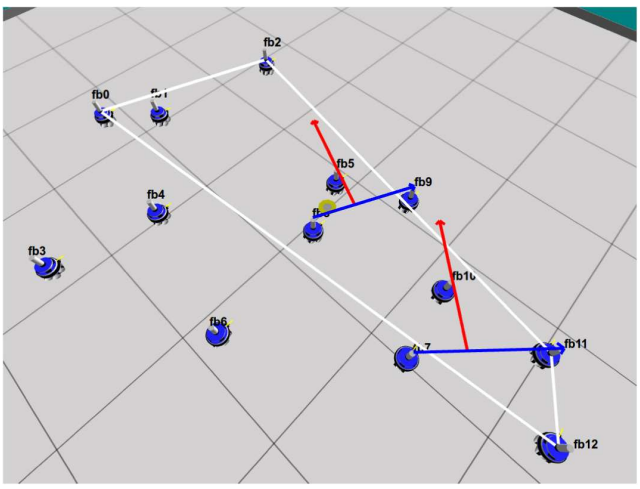

FIGURE 8. System reconfiguration stage: a new coordinate system is built with the help of four marginal robots.

The centroid of this tetragon is marked with a green circle, and the robot nearest to this point is selected as a new leader, i.e. robot `fb5`. Using the same strategy as in the initialization stage, robots `fb8` and `fb9` become reference robots, which together form a new coordinate system. Once this coordinate

system is propagated to all the robots within the multi–robot team, the old coordinate system is dismissed.

B. VALIDATION OF DYNAMIC LOCALIZATION

As presented in the previous sections, the EKF estimator can be applied for dynamic localization. In this subsection, this approach is validated with Matlab, having the following assumptions: a) all robots move randomly, b) the measurements are associated with the white Gaussian noise to simulate the noise of the real sensor, and finally c) the entire simulation (including measurements) are updated at 10 Hz.

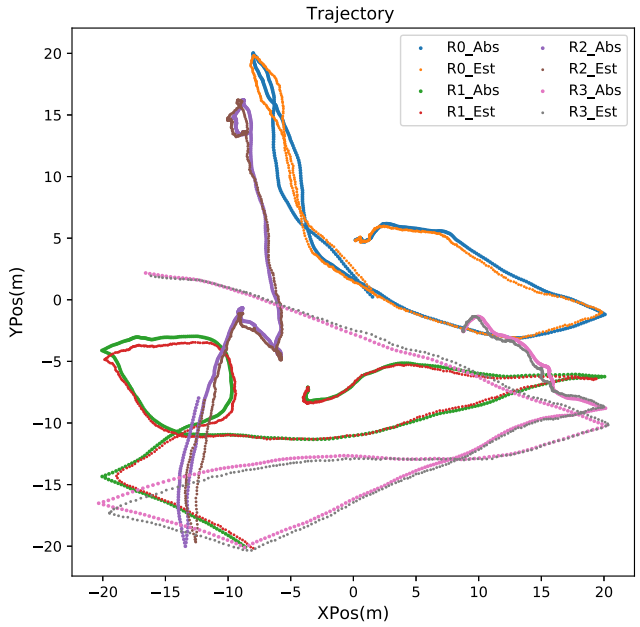

FIGURE 9. Localization of four robots with the EKF.

In a first experiment, we place four robots randomly in an area of 40×40 m^2. The ground-truth trajectory of each robot, and its estimation are shown in Fig. 9, while the localization error and covariance matrix determinant of the EKF are given in Fig. 10. To statistically evaluate our results, we run 30 simulations and acquire around 72000 estimations. We divide the localization error into 25 0.1-meter bins and we present the histogram of the error in Fig. 11 and note that for more than half of our samples, the error falls into the bins around 0.3 meters.

In our simulation, we assume that the range measurements arrive simultaneously at each time step. However, in a distributed system, it is very challenging to achieve such synchronization. This was analyzed by many researchers to provide performance limits [43] and solutions [44]. Other approaches, such as Unscented Kalman Filters, can also be applied to reduce the impact of poor synchronization. Since all of these approaches can be applied in our system to improve the estimation accuracy and mitigate the impact of asynchronism, we plan to address the issue in future work.

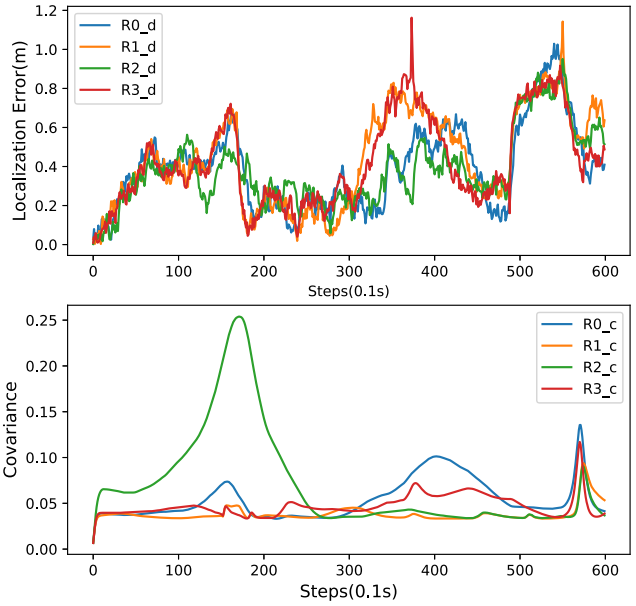

FIGURE 10. Localization error and covariance trend.

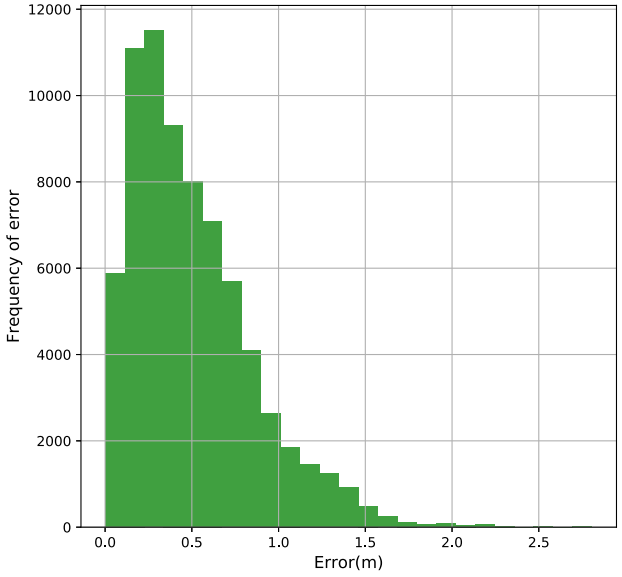

FIGURE 11. Histogram of the localization error over thirty experimental runs.

VI. CONCLUSION

This work presents a dynamic localization approach based on range–only UWB sensors. By using a bio–inspired decentralized consensus technique, our approach leverages the communication between robots to build a common coordinate system which allows for localization within an unknown region, with only three robots. Therefore, to the best of our knowledge, the presented work offers an advantage to existing approaches which require static robots which are used for localization. Furthermore, we developed an algorithm used for the initialization, and an additional algorithm which is

used for the reconfiguration phase. During the initialization, for a brief time instance, robots which are selected by consensus are static, create a common coordinate system and propagate it to all other members of a multi–robot team. After this, a second algorithm, continuously, dynamically and automatically improves the localization by re–initializing the common coordinate system. The great advantage of this approach is that in the localization phase, it doesn't require robots to be static as it uses the EKF for pose estimation.

Through simulations in Matlab and ARGoS, we have validated and demonstrated that the EKF can be successfully applied in for the aforementioned purpose.

We plan further investigation into the error characteristics of UWB and robot formations to extend the presented method to support a variety of robot distributions. We believe we can achieve this by selecting a more appropriate algorithm to determine the marginal robots, as the current approach can be greatly improved from the standpoint of the necessary data exchange. Also, our goal is to implement the method on physical robots and consider different dynamic and complex environments.

ACKNOWLEDGEMENTS

The authors would like to thank H. Benzerrouk for the discussions on the EKF model, and V. S. Varadharajan and J. Panerati, for their help with the ARGoS simulator and the Buzz programming language.

REFERENCES

[1] M. S. Couceiro, "An overview of swarm robotics for search and rescue applications," in *Handbook of Research on Design, Control, and Modeling of Swarm Robotics*. Hershey, PA, USA: IGI Global, 2016, pp. 345–382.
[2] M. Brambilla, E. Ferrante, M. Birattari, and M. Dorigo, "Swarm robotics: A review from the swarm engineering perspective," *Swarm Intell.*, vol. 7, no. 1, pp. 1–41, Mar. 2013. [Online]. Available: https://link.springer.com/article/10.1007/s11721-012-0075-2
[3] J. J. Leonard and H. F. Durrant-Whyte, "Mobile robot localization by tracking geometric beacons," *IEEE Trans. Robot. Autom.*, vol. 7, no. 3, pp. 376–382, Jun. 1991.
[4] V.-L. Dang, B.-S. Le, T.-T. Bui, H.-T. Huynh, and C.-K. Pham, "A decentralized localization scheme for swarm robotics based on coordinate geometry and distributed gradient descent," in *Proc. MATEC Web Conf.*, vol. 54, 2016, p. 6.
[5] P. Skrzypczy ski, "Mobile robot localization: Where we are and what are the challenges?" in *Proc. Int. Conf. Automat.* Cham, Switzerland: Springer, 2017, pp. 249–267.
[6] J. Borenstein, H. R. Everett, and L. Feng, "Where am I? Sensors and methods for mobile robot positioning," *Univ. Michigan*, vol. 119, no. 120, p. 27, 1996.
[7] G. A. Korikar, S. K. Katragadda, S. Kesarla, J. M. Conrad, and A. F. Browne, "A survey on robot localization in extraterrestrial environments," in *Proc. SoutheastCon*, 2016, pp. 1–7.
[8] I. Conduraru and I. Doroftei, "Localization methods for mobile robots—A review," in *Proc. Adv. Mater. Res.*, vol. 837, 2014, pp. 561–566.
[9] J. Torres-Solis, T. H. Falk, and T. Chau, "A review of indoor localization technologies: Towards navigational assistance for topographical disorientation," in *Ambient Intelligence*. Rijeka, Croatia: InTech, 2010.
[10] J. Wang and E. Olson, "AprilTag 2: Efficient and robust fiducial detection," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2016, pp. 4193–4198.
[11] T. Lochmatter, P. Roduit, C. Cianci, N. Correll, J. Jacot, and A. Martinoli, "Swistrack—A flexible open source tracking software for multi-agent systems," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2008, pp. 4004–4010.
[12] Natural Point. (2011). *Optitrack*. Accessed: Feb. 22, 2014. [Online]. Available: http://www.naturalpoint.com/optitrack/
[13] V. Peak, "Vicon motion capture system," Tech. Rep., 2005.
[14] A. Prorok and A. Martinoli, "Accurate indoor localization with ultra-wideband using spatial models and collaboration," *Int. J. Robot. Res.*, vol. 33, no. 4, pp. 547–568, 2014.
[15] I. Guvenc, S. Gezici, and Z. Sahinoglu, "Ultra-wideband range estimation: Theoretical limits and practical algorithms," in *Proc. IEEE Int. Conf. Ultra-Wideband (ICUWB)*, vol. 3, Sep. 2008, pp. 93–96.
[16] J. González *et al.*, "Mobile robot localization based on ultra-wide-band ranging: A particle filter approach," *Robot. Auton. Syst.*, vol. 57, no. 5, pp. 496–507, 2009.
[17] G. A. Hollinger, J. Djugash, and S. Singh, "Target tracking without line of sight using range from radio," *Auton. Robots*, vol. 32, no. 1, pp. 1–14, 2012.
[18] A. Prorok and A. Martinoli, "A reciprocal sampling algorithm for lightweight distributed multi-robot localization," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Sep. 2011, pp. 3241–3247.
[19] A. Prorok, A. Bahr, and A. Martinoli, "Low-cost collaborative localization for large-scale multi-robot systems," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, May 2012, pp. 4236–4241.
[20] A. Bahr, J. J. Leonard, and M. F. Fallon, "Cooperative localization for autonomous underwater vehicles," *Int. J. Robot. Res.*, vol. 28, no. 6, pp. 714–728, 2009.
[21] E. Liao, G. Hollinger, J. Djugash, and S. Singh, "Preliminary results in tracking mobile targets using range sensors from multiple robots," in *Distributed Autonomous Robotic Systems 7*. Tokyo, Japan: Springer, 2006, pp. 125–134.
[22] W. Zhang, J. A. Djugash, and S. Singh, "Parrots: A range measuring sensor network," Robot. Inst., Carnegie Mellon Univ., Pittsburgh, PA, USA, Tech. Rep. CMU-RI-TR-06-05, 2006. [Online]. Available: http://repository.cmu.edu/robotics/966/
[23] R. Kurazume, S. Nagata, and S. Hirose, "Cooperative positioning with multiple robots," in *Proc. IEEE Int. Conf. Robot. Automat.*, May 1994, pp. 1250–1257.
[24] A. Cornejo and R. Nagpal, "Distributed range-based relative localization of robot swarms," in *Algorithmic Foundations of Robotics XI*. Cham, Switzerland: Springer, 2015, pp. 91–107.
[25] V. Indelman, E. Nelson, J. Dong, N. Michael, and F. Dellaert, "Incremental distributed inference from arbitrary poses and unknown data association: Using collaborating robots to establish a common reference," *IEEE Control Syst.*, vol. 36, no. 2, pp. 41–74, Apr. 2016.
[26] A. R. Shirazi and Y. Jin, "A strategy for self-organized coordinated motion of a swarm of minimalist robots," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 1, no. 5, pp. 326–338, Oct. 2017.
[27] H. Wymeersch, J. Lien, and M. Z. Win, "Cooperative localization in wireless networks," *Proc. IEEE*, vol. 97, no. 2, pp. 427–450, Feb. 2009.
[28] I. M. Rekleitis, G. Dudek, and E. E. Milios, "Multi-robot cooperative localization: A study of trade-offs between efficiency and accuracy," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, vol. 3, Sep. 2002, pp. 2690–2695.
[29] D. Fox, W. Burgard, H. Kruppa, and S. Thrun, "Collaborative multi-robot localization," in *Mustererkennung 1999* (Informatik aktuell). Berlin, Germany: Springer, 1999, pp. 15–26, doi: 10.1007/978-3-642-60243-6_2.
[30] S. Thrun, "Probabilistic robotics," *Commun. ACM*, vol. 45, no. 3, pp. 52–57, 2002.
[31] A. Prorok, A. Bahr, and A. Martinoli, "Low-cost multi-robot localization," in *Redundancy in Robot Manipulators and Multi-Robot Systems*. Berlin, Germany: Springer, 2013, pp. 15–33.
[32] L. Luft, T. Schubert, S. I. Roumeliotis, and W. Burgard, "Recursive decentralized collaborative localization for sparsely communicating robots," in *Robotics: Science and Systems*. New York, NY, USA, 2016.
[33] C. Pinciroli, A. Lee-Brown, and G. Beltrame, "A tuple space for data sharing in robot swarms," in *Proc. 9th EAI Int. Conf. Bio-Inspired Inf. Commun. Technol. (BIONETICS)*, 2016, pp. 287–294.
[34] C. Pinciroli and G. Beltrame, "Buzz: An extensible programming language for heterogeneous swarm robotics," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2016, pp. 3794–3800.
[35] G. MacGougan, K. O'Keefe, and R. Klukas, "Ultra-wideband ranging precision and accuracy," *Meas. Sci. Technol.*, vol. 20, no. 9, p. 095105, 2009.
[36] H. Benzerrouk, A. Nebylov, S. Hassen, and P. Closas, "≪Cubature & Gauss-hermite≫ based Kalman filters applied to unmanned aerial vehicle attitude estimation problem," *J. Moscow Aviation Inst.*, vol. 20, no. 5, 2013.

[37] P. Müller and R. Piché, "Statistical trilateration with skew-t errors," in *Proc. Int. Conf. Localization GNSS (ICL-GNSS)*, 2015, pp. 1–6.
[38] R. E. Kalman and R. S. Bucy, "New results in linear filtering and prediction theory," *J. Basic Eng.*, vol. 83, no. 1, pp. 95–108, 1961.
[39] R. S. Bucy, "Linear and nonlinear filtering," *Proc. IEEE*, vol. 58, no. 6, pp. 854–864, Jun. 1970.
[40] A. H. Jazwinski, *Stochastic Processes and Filtering Theory*. North Chelmsford, MA, USA: Courier Corporation, 2007.
[41] P. S. Maybeck, *Stochastic Models, Estimation, and Control*. New York, NY, USA: Academic, 1979.
[42] C. Pinciroli *et al.*, "ARGoS: A modular, parallel, multi-engine simulator for multi-robot systems," *Swarm Intell.*, vol. 6, no. 4, pp. 271–295, 2012.
[43] Y. Xiong, N. Wu, Y. Shen, and M. Z. Win, "Cooperative network synchronization: Asymptotic analysis," *IEEE Trans. Signal Process.*, vol. 66, no. 3, pp. 757–772, Feb. 2018.
[44] W. Yuan, N. Wu, B. Etzlinger, H. Wang, and J. Kuang, "Cooperative joint localization and clock synchronization based on Gaussian message passing in asynchronous wireless networks," *IEEE Trans. Veh. Technol.*, vol. 65, no. 9, pp. 7258–7273, Sep. 2016.

YANJUN CAO received the master's degree in mechatronic engineering from the Shenyang Institute of Automation, Chinese Academy of Sciences, Shenyang, China, in 2015. He is currently pursuing the Ph.D. degree with the Research Laboratory for Making Innovative Space Technology, Department of Computer and Software Engineering, Polytechnique Montréal. He has been developing a novel kind of modular robot named 360Bot (two generations) since 2010. His research interests mainly focus on multi-robot systems, including self-reconfigurable modular robots and swarm robotics, simultaneous localization and mapping for swarm robotics, and mobile robot exploration.

MENG LI received the B.E. and M.S. degrees in electronic engineering from the Beijing University of Aeronautics and Astronautics, Beijing, China, in 2004 and 2007, respectively, and the Ph.D. degree in electrical engineering from Polytechnique Montréal, Montreal, QC, Canada, in 2016. Since 2016, he has been with the Department of Electrical Engineering, Polytechnique Montréal, as a Post-Doctoral Fellow. Since 2017, he has been with the Research Laboratory for Making Innovative Space Technology, Department of Computer and Software Engineering, Polytechnique Montréal, as a Post-Doctoral Researcher Fellow. He is currently an Associate Professor with the School of Information and Science Technology, Zhejiang Sci-Tech University, Hangzhou, China. His research interests include mobile robot localization, swarm robot control, communication networks, task scheduling, fault tolerance, parallel computing, and real-time systems.

IVAN VOGOR received the Ph.D. degree in information and software engineering from the University of Zagreb, Croatia. From 2012 to 2014, he was an External Graduate Student at Mälardalen University, Sweden, involved in component-based software engineering techniques for underwater robots. During that time, he has successfully participated multiple student competitions, most notably AUVSI RoboSub in San Diego, USA. As a Graduate Student, he was a Teaching Assistant and developed materials for multiple courses, as well as a collaborator on several EU funded. From 2017 to 2018, he was a Post-Doctoral Research Fellow at the MIST Lab, Polytechnique Montréal, Canada. He is currently a Senior Robotics Software Engineer with Gideon Brothers, Croatia. His research interests include software architecture for robotics, cyber-physical systems, swarm robotics, and green software.

SHAOMING WEI received the B.S. and Ph.D. degrees in electronic and information engineering from the Beijing University of Aeronautics and Astronautics, Beijing, China, in 2007 and 2013, respectively. He is currently a Lecturer with the School of Electronic and Information Engineering, Beihang University. His research interests include high-resolution radar signal processing, 3-D reconstruction, and multi-target tracking.

GIOVANNI BELTRAME received the Ph.D. degree in computer engineering from Politecnico di Milano in 2006. He was a Microelectronics Engineer with the European Space Agency on a number of projects spanning from radiation-tolerant systems to computer-aided design. In 2010, he moved to Polytechnique Montréal, Montreal, Canada, where he is currently an Associate Professor with the Computer and Software Engineering Department and the Director of the Research Laboratory for Making Innovative Space Technology, Polytechnique Montréal, with more than 20 students and postdoctoral researchers under his supervision. He is also a Visiting Professor with the University of Tübingen. His research interests include modeling and design of embedded systems, artificial intelligence, and robotics. He received over 15 grants from government agencies and industry.